For this assignment 7, I made it so users are able to have an engaging experience when playing the Archaeology game. For this, I've placed a grid-based layout including CSS. I chose for there to be a textbox as an option, but still users are able to go into the developers console if they choose. I then made sure alerts were not popping up upon game loading as well as throughout it. However, a message will show below the game display when the correct cell is chosen. Below I will explain snippets of code along with explanations for it. I'm going to explain my JS code, text-input box and event driven response. The user's input triggers and initializes the 'tryDig' function.

First two variables: These two variables shown here attempts and ruinsExcavated are declared and meant to keep track of user's tries in the games as well as the ruins that are excavated successfully.

```
let attempts = 0; let ruinsExcavated = 0;
```

Attempts counter: Helps to keep track of the user's efforts in the game. And the targetObj stores results of excavation on the board using dig

```
attempts++;
let targetObj = board.dig(targetCell);
```

If statement: This statement was used to verify if digging resulted in something. If something is found, the code is executed.

```
if(targetObj) {
```

Showing the digging output: This line of code is meant to show a success message for the user. The HTML element with the ID: 'messageDisplay' is retrieved. This syntax then updates the message displayed on the screen.

```
document.getElementById('messageDisplay'). textContent='Looks like you found a' + targetObj,name;
```

Appearance of successful digs: These lines of codes explain the symbol in the cells' appearance

```
targetCell.innerHTML= '©';  - innerHTML property changes the content of the cell to display the symbol-
targetCell.style.color = 'pink'; -style.color property is used to change the text color to pink from black-
targetCell.classList.add('yes'); -classList.add method adds the class 'yes' for the cell & correlates styling in CSS-
ruinsExcavated += 1;  -adding one to subtotal of good digs-
```

If statement: This statement checks on the amount of good digs matching the number of objects to be dug
Summary message: If this condition results in true then a message shows the success from excavating.

```
if(targetObj.successes === targetObj,size) {
document.getElementbyId('summaryMessage').textContent = 'An excavated' + targetObj,name;
```

Else statement: If condition is not met this part of the code changes the message display to show the dig unsuccessful. The targeted cell then shows a different symbol such as ¥. I chose for there to be an empty string/message so nothing shows exactly how nothing shows when you click an empty cell.

```
else {
document.getElementById('messageDisplay').textContent = ' ';
targetCell.innerHTML= ¥;  - innerHTML property changes the content of the cell to display the symbol-
targetCell.style.color = purple; -style.color property is used to change the text color to purple from black-
targetCell.classList.add(no); -classList.add method adds the class no for the cell & correlates styling in CSS-
```

Updating user progress: I called this function solely for updating performance display based on the users efforts and their progress. Only after the digging is this updated along with attempts made and a summary message.

```
  }
updatePerformanceDisplay();
}
```

Retrieved variables: HTML elements with each ID has a variable that is assigned. The variable is in parentheses as shown below.

```
let attempts = document.getElementById('attempts');
let ruinsExcavated=document.getElementById(ruinsExcavated');
let summaryMessages=document.getElementById('summaryMessage');
```

If statement: checks that the subtotal amount of digs is equal to amount on the board. If this is true t
then a message is shown that positively reinforces the user; “Hey, look at you, that’s everything! Nice operating.” As for the triple equal sign, Craig helped me to remember from explaining in lecture that if boolean expression is what you want you use triple equal sign. So the triple equal sign checks for both value and type whereas double checks for value. So for this case I want accurate comparison between ruinsExcavated and board.ruins.length.

```
if(ruinsExcavated ===board.ruins.length) (
    summaryMessage.textContent= “Hey, look at you, that’s everything! Nice operating.”
```

Assigning function : I used onchange as an event of the element with the ID coordinates here

```
document.getElementById(“coordinates”).onchange = function(event) {
```

Creating variables: Variable was created, ‘cellId’ and the value was set to the event.target value property

```
let cellId = event.target.value; Another variable created, ‘targetCell’ with value of ‘cell’ concatenated to ‘cellId’
let targetCell = document.getElementById(‘cell” + cellId);
```

Calling tryDig function: targetCell passed as the argument. The function gets external input to perform operations.

```
tryDig(targetCell);
```

Created variable: The variable created here is cell of course. The value is set to the b6 cell. So the game may start with this. Just refresh if necessary.

```
let cell =document.querySelector(#cellb6’);
```

Calling function: and of course tryDig(cell); here I am calling the function with cell as the function.